# MASTER'S THESIS

## Tekst en association rule mining voor detecteren van workarounds in vrije tekst die gestructureerde data-invoer omzeilen

van Rouwendal, J. (Jan)

**Award date:**
2020

# Tekst en association rule mining voor detecteren van workarounds in vrije tekst die gestructureerde data-invoer omzeilen


| | |
|---|---|
| Opleiding: | Open Universiteit, faculteit Management, Science & Technology<br>Masteropleiding Business Process Management & IT |
| Cursus: | IM0602 Voorbereiden Afstuderen BPMIT<br>IM9806 Afstudeertraject Business Process Management and IT |
| Student: | Jan van Rouwendal |
| Identiteitsnummer: | |
| Datum: | 26-1-2020 |
| Afstudeerbegeleider | Dr. Lloyd Rutledge |
| Meelezer | Dr. Guy Janssens |
| Versie nummer: | 9.8 |
| Status: | definitief |

# Abstract

Veel informatiesystemen hebben functionele beperkingen om structurele data in te voeren en gebruikers kunnen dit omzeilen door vrije tekst in het informatiesysteem te gebruiken als workaround. Deze workarounds zijn een onderdeel van shadow-IT en gebruikers passen dit toe zonder bewustwording, acceptatie of ondersteuning van de organisatie. Detectie van dit fenomeen is daarom lastig, terwijl het juist inzicht geeft hoe we de functionaliteiten of processen van het systeem kunnen verbeteren en daarmee de weerbaarheid en veerkracht (resilient) verhogen. De toename in data science technologie biedt nieuwe mogelijkheden voor deze detectie door met tekstmining (TM) kenmerken te extraheren uit de teksten en met association rule mining (ARM) algoritmes de relaties te leggen naar omzeilde data. In deze thesis onderzoeken we de detectiekwaliteit van TM en de Apriori, FPGrowth en Tertius ARM-algoritmes gecombineerd met de voorspellingswaarde van potentieel omzeilde data. We dragen bij aan het literatuuronderzoek van workarounds door te constateren dat TM significante referentiewoorden van de workaround in tekst lokaliseert en ARM deze relateert aan de omzeilde gestructureerde data-invoer. Met verdiepende tekstmanipulatie- en aggregatietechnieken zijn de workarounds betrouwbaar te detecteren in combinatie met taalkundige analyse. Voor de praktijk hebben we een stappenplan voor detectie opgesteld en geven we aanbevelingen voor extern validatie met verbreding in tekstmanipulatie- en aggregatietechnieken om het automatische proces te verbeteren.



Sleutelbegrippen
shadow-IT, workarounds, tekstmining, text mining, association rule mining, resilience mining

# Inhoudsopgave

# 1. Introductie

## 1.1. Achtergrond

De meest gebruikte definitie voor shadow-IT is alle hardware, software of elke andere oplossing gebruikt door personeel zonder bewustwording, acceptatie, kennis of ondersteuning van de IT-afdeling. Het toepassen van workarounds binnen een informatiesysteem is één van de 6 gebieden van shadow-IT die primair gericht is op misbruik van bestaande systemen om beperkingen van het systeem te ontwijken, zodat gebruikers hun werk kunnen doen (Kopper & Westner, 2016). Het zijn aanpassingen, improvisaties of andere wijzigingen in een systeem om obstakels, bijzonderheden, gebruiken, onregelmatigheden, misverstanden, verwachtingen en beperkingen op te lossen, met als doel om effectiviteit en efficiëntie van de organisatie en persoonlijke resultaten te bereiken (Alter, 2014). De consequentie hiervan is het ontstaan van inaccurate of verborgen data wat kan leiden tot inefficiëntie, fouten en security bedreigingen (Kopper & Westner, 2016) en de positieve keerzijde is dat het detecteren van workarounds als aanpak kan dienen voor het verbeteren van het informatiesysteem (Alter, 2014).

De Resilience Mining afstudeerkring onderzoekt het detecteren van workarounds om de weerbaarheid en veerkracht (resilient) van een informatiesysteem te verbeteren met clustering, classification, outlier detection en association rule mining algoritmes gescheiden in de datasoorten vrije tekst en gestructureerde data. Ons deelonderzoek is gericht op de toepasbaarheid van tekstmining (TM) en association rule mining (ARM) algoritmes om te detecteren of gebruikers gestructureerde data invoeren in vrije tekst, omdat de daarvoor bedoelde invoer in het systeem te lastig of te ingewikkeld is. Het doel is dat IT-managers met deze detectie verbeteringen doorvoeren in de vorm van systeem- en procesveranderingen.

## 1.2. Gebiedsverkenning

Shadow-IT is een groeiend onderzoeksgebied gedreven door de negatieve aspecten zoals beveiliging, compliance en inefficiëntie en ook positieve effecten zoals technologische innovatie en flexibiliteit. Workarounds is een concept in dit onderzoeksgebied tezamen met feral practices, shadow systems, un-enacted projects en shadow sourcing (Kopper & Westner, 2016).

Workarounds is een gebruikelijk thema bij grotere systemen die ontworpen zijn voor geformaliseerde processen waarbij gebruikers weerstand hebben, omdat ze delen van het systeem als ongeschikt ervaren voor hun werk en de IT-functionaliteiten beperkt of inefficiënt zijn of niet aansluiten op de gangbare routines (Kopper & Westner, 2016). De invoer- en overzichtsschermen van de meeste systemen zijn suboptimaal en kunnen leiden tot aanpassingen en improvisaties door de gebruiker, omdat ze data niet op een natuurlijke wijze kunnen invoeren om hun werk gedaan te krijgen (Collins, Fred, Wilcox, & Vawdrey, 2012). Een improvisatie is het gebruiken van vrije tekst in plaats van de gestructureerde data-invoer en komt voor in situaties waarbij gebruikers te veel moeten klikken, ze complexe structurele invoer van data willen omzeilen, ad-hoc een snelle registratie willen verrichten, voorkeur voor een tekstoverzicht hebben of datafragmentatie in een systeem willen voorkomen. Deze workaround vraagt geen bijzondere vaardigheden van de gebruiker en het effect is dat de data inaccuraat, onvindbaar en niet voor analyse beschikbaar is (Huuskonen & Vakkari, 2013).

Het omzeilen van gestructureerde data-invoer kan ontstaan door onbekendheid met het systeem en dit vraagt om investeringen in kennis of sturing in gedrag bij de gebruikers (Huuskonen & Vakkari, 2013). In stressvolle situaties met snelle acties is zo'n workaround goedgekeurd en is het een creatieve oplossing om het werk gedaan te krijgen (Alter, 2014). In de situatie dat het een routinematige handeling is geworden wegen de efficiency voordelen op tegen de overtreding, waarbij gebruikers zich bewust zijn van de consequenties (Röder, Wiesche, & Schermann, 2014). Het ontwijken of goedkeuren van de workaround is een optie als het omzeilen onschadelijk is voor de organisatie en in de andere gevallen verdient herontwerp van het proces of systeem de voorkeur boven het voorkomen (Van de Weerd, Vollers, Beerepoot, & Fantinato, 2019). Voor herontwerp geeft detectie van de workaround routine met herleiding van de motivatie inzicht in de ontwerpverbeteringen van het systeem (Röder, Wiesche, & Schermann, 2014).

In het onderzoeksgebied van Shadow-IT is met data science detectiemethodes organisatorisch onderzoek verricht (Fürstenau & Rothe, 2014) en biedt TM als onderdeel van data science nieuwe mogelijkheden voor verkennende en theorie gedreven onderzoek (Kobayashi, Mol, Berkers, Kismihok, & Den Hartog, 2018). Het frequent voorkomen van dezelfde inhoud of herkenbare verwijzingen in tekst is een detectiekenmerk dat gestructureerde data-invoer is omzeilt (Huuskonen & Vakkari, 2013) en TM extraheert efficiënt en betrouwbaar databewerkingen, herhalingen en signalen uit tekst. Voor het analyseren van de samenhang in data zijn algoritmes zoals classificatie, ARM, clustering, topic modelling, similarity & distance computing en dimensionality reduction toepasbaar (Kobayashi, Mol, Berkers, Kismihok, & Den Hartog, 2018).

Het detecteren van deze workaround valt onder een unsupervised learning probleem om onbekende patronen tussen data te herleiden, waarbij clustering en ARM-algoritmes geëigende methoden zijn om deze op te lossen. ARM is de meest geschikt vorm, omdat het associatieregels genereert, die de relaties tussen een workaround in de tekst en de omzeilde data aantoont en herbruikbare detectie kennis geeft voor verdere analyse (Kulkarni & Kulkarni, 2016).

Voor wetenschappelijk onderzoek zijn algoritmes inzetbaar om hypotheses te testen, waarbij de onderzoeksprocessen het verzamelen, schonen en transformeren van data doorlopen om vervolgens de patronen en relaties te interpreteren, evalueren en valideren (Kobayashi, Mol, Berkers, Kismihok, & Den Hartog, 2018).

## 1.3. Probleemstelling

Workarounds waarbij gebruikers gestructureerde data-invoer omzeilen met vrije tekst en de invoerfunctionaliteit wel aanwezig is, zijn een indicatie dat het ontwerp van de schermen inefficiënt is, waarbij detectie inzicht geeft in het noodzakelijke herontwerp (Huuskonen & Vakkari, 2013). De toepasbaarheid van TM en ARM-algoritmes voor detectie van deze workarounds vormt de hoofdvraag van de thesis:

"Hoe en in welke mate zijn tekstmining (TM) en association rule mining (ARM) algoritmes toepasbaar om in vrije tekst workarounds te detecteren, waarmee gebruikers gestructureerde data-invoer binnen het informatiesysteem omzeilen".

Ons onderzoek is gericht op het detecteren van een relatie tussen ingevoerde tekst en omzeilde data, zodat we de *omzeilde workaround data* in de tekst voorspellen. Daarvoor definiëren we op het gebied van literatuur en techniek de volgende deelvragen:

a) *Hoe toepasbaar is TM voor het detecteren ingevoerde tekst die data omzeilt?*
b) *Hoe toepasbaar is ARM om een relatie naar omzeilde data te detecteren?*
c) *Welke kennis geven de relaties tussen ingevoerde tekst en omzeilde data?*

## 1.4. Motivatie en relevantie

Workarounds zijn inherent verbonden aan het gebruik van informatiesystemen en zijn als onderdeel van shadow-IT een achtergebleven onderzoeksgebied (Alter, 2014). IT-managers zijn bekend met workarounds (Kopper & Westner, 2016) en zijn beter af om deze gecentraliseerd aan te pakken met standaardprocedures om deze te detecteren (Fürstenau & Rothe, 2014). De meeste onderzoeken zijn gericht op inventarisatie van de verschillende vormen, het gedrag en de motivatie van workarounds en geven een beperkt inzicht in detectiemethodes (Huuskonen & Vakkari, 2013). Voor efficiënte en gecentraliseerde detectie van shadow-IT zijn al data science algoritmes toegepast (Fürstenau & Rothe, 2014) en onderzoeken naar detectie van workarounds ontbreken, terwijl adequate algoritmes daarvoor beschikbaar zijn (Kobayashi, Mol, Berkers, Kismihok, & Den Hartog, 2018).

Een workaround in vrije tekst met omzeilde data-invoer is een optimaal fenomeen voor verkenning met data science techniek, omdat er bij een vrije tekst detecteerbare herkenningspunten zijn met een relatie naar omzeilde data (Huuskonen & Vakkari, 2013) en ze routinematig voorkomen, zodat ze geschikt zijn voor automatische analyse (Röder, Wiesche, & Schermann, 2014). Detectie binnen teksten is haalbaar omdat TM een adequaat instrument is voor organisatorisch onderzoek (Kobayashi, Mol, Berkers, Kismihok, & Den Hartog, 2018) en ARM detectieregels genereert voor hergebruik in de organisatie (Kulkarni & Kulkarni, 2016).

## 1.5. Aanpak in hoofdlijnen

In het tweede hoofdstuk verrichten we een literatuuronderzoek voor het theoretisch kader met vragen over workarounds, omzeilde data, detectiemethodieken en TM en ARM-technologie. We onderbouwen het conceptueel model met de selectie van de casus en het technisch ontwerp voor het uitvoeren van het onderzoek in het derde hoofdstuk over methodologie. De resultaten van de experimenten zijn opgenomen in het vierde hoofdstuk, waarbij we afsluiten in een hoofdstuk met de discussie, reflectie, conclusies en aanbevelingen voor vervolgonderzoek op techniek en extern validatie-onderzoek (Saunders, Lewis, & Thornhill, 2016).

# 2. Theoretisch kader

In ons literatuuronderzoek krijgen we inzicht in de wetenschappelijke kennis van workarounds in teksten en welke methodieken en modellen daarvoor beschikbaar zijn op het gebied van TM en ARM. We onderscheiden de volgende vragen:

- Wat zijn de kenmerken van workarounds in tekst die data omzeilen?
- Welke methodieken voor detecteren van omzeilde data zijn er en zijn deze bruikbaar?
- Hoe is TM en ARM toepasbaar om omzeilde data in tekst te voorspellen?
- Welke technieken van ARM-algoritmes zijn beschikbaar voor detectie?

De antwoorden op deze vragen gebruiken we voor het ontwerp van het onderzoek om de hoofdvraag te beantwoorden.

## 2.1. Onderzoek aanpak

Om een gedegen kennis en inzicht in de trends te verkrijgen zijn artikelen op het gebied van shadow-IT onderzocht gericht op de definitie van workarounds als taxonomie van shadow-IT (Kopper & Westner, 2016), waarna we onderzoeken over workarounds op kenmerken van misbruik in vrije tekst in relatie tot gestructureerde data-invoer hebben bestudeerd. Door literatuur over shadow-IT te onderzoeken op detectievormen met data science (Fürstenau & Rothe, 2014) is inzicht verkregen in de toepasbaarheid van metadata van het datamodel. Onderzoeken buiten shadow-IT zijn geanalyseerd op methodieken voor tekst- en relatie-analyse (Kobayashi, Mol, Berkers, Kismihok, & Den Hartog, 2018) en op de modellen voor de TM en ARM-algoritmes (Kulkarni & Kulkarni, 2016). Aansluitend is een verdiepende studie verricht naar de beschikbaarheid van de algoritmes in de technische platformen (Arora, Shelza, & Rao, 2013).

De voor het theoretische framework onderzochte artikelen zijn peer-reviewed en afkomstig uit wetenschappelijke tijdschriften en conferenties (Saunders, Lewis, & Thornhill, 2016).

## 2.2. Workaround

De vraag *wat zijn de kenmerken van workarounds in tekst die data omzeilen* is beantwoord met de volgende literatuur.

In een overzichtsonderzoek naar de taxonomie over shadow-IT zijn door Kopper en Westner (2016) 58 artikelen bestudeerd op het gebied shadow-IT waarvan 8 artikelen over de taxonomie van workarounds gaan. Een groot deel daarvan is gericht op voorkoming van workaround gedrag in de gezondheidszorg, mede doordat de privacy borging van data in deze branche essentieel is. Deze onderzoeken beschrijven verschillende vormen van workarounds in tekst en ook de vorm van omzeilen van gestructureerde data (Huuskonen & Vakkari, 2013).

Als gebruikers data in vrije tekst invoeren in plaats van gebruik te maken van de bedoelde invoerschermen, willen ze deze data als essentieel markeren en daarmee onderscheiden van minder relevante data in de tekst. Om relaties te leggen naar waar de data oorspronkelijk moet zijn ingevoerd maken gebruikers referentiewoorden aan in deze tekst. Zo ontstaan ongeschreven regels binnen de organisatie voor het uitvoeren van deze vorm van workaround, waardoor de structurele data *leeg* blijft en niet bruikbaar is. Het gedrag bij deze vrije tekst invoer toont individuele herhalingen, waarbij het frequent voorkomen van dezelfde

inhoud in tekst kenmerken zijn van deze omzeiling. Omdat tekst een gebrek aan structuur heeft, maken gebruikers zelf een structuur aan met signaalinformatie die combinaties bevatten van hoofdletters, tekens, verwijswoorden, afkortingen en lijsten (Huuskonen & Vakkari, 2013).

De in de onderzoeken gevonden referentiewoorden en signaalinformatie zijn kenmerken van een workaround, waarmee de gebruiker data omzeilt en kunnen we in ons onderzoek herleiden. De frequentie van de woorden en signaalinformatie is toepasbaar als een onderscheidende factor voor detectie. Ook is de relatie naar het *leeg* zijn van de omzeilde data een kenmerk van deze workarounds, waarbij we het ontbreken van deze data een criteria is.

## 2.3. Methodieken

Op de vraag *welke methodieken voor detecteren van omzeilde data zijn er en zijn deze bruikbaar* geeft de wetenschap onderstaande methode om shadow-IT te detecteren en te evalueren.

Voor het detecteren van shadow-IT binnen een organisatie is een procedureel model beschikbaar wat ook van toepassing is op het deelgebied workarounds. De eerste stap is de project-initiatie gevolgd door datacollectie met aansluitende data-analyse, waarna implementatie van maatregelen volgt ter voorkoming van het fenomeen. Het advies bij de datacollectie is om een template-aanpak te gebruiken wat is gerelateerd aan het gestructureerde datamodel.

De template-aanpak is gebaseerd op het inventariseren van de potentiele vormen van shadow-IT, waarbij we *kandidaten* identificeren door middel van het opvragen van data uit het systeem. Door een semi-automatische analyse van *kandidaten* met expert kennis, detecteren we oneigenlijk gebruik van de systemen. Ze geven de aanbeveling om deze methode in combinatie met relatie-analyses toe te passen voor vormen van shadow-IT en deze processen te automatiseren (Fürstenau & Rothe, 2014)

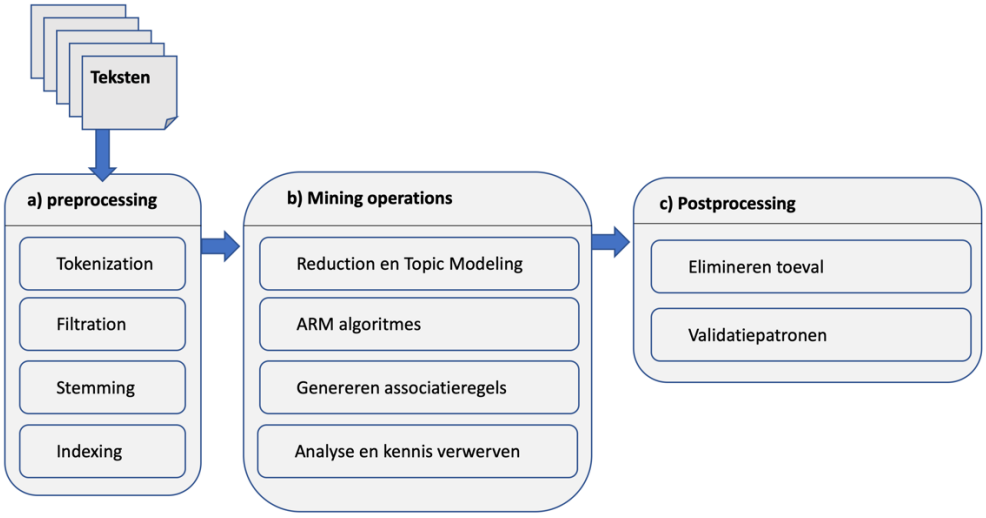
Alhoewel deze methode gebruikt is op een macroniveau van systemen is dezelfde methodiek bruikbaar op een microniveau binnen één systeem, omdat de analyse gericht is op één of meerdere gestructureerde datamodellen. Door het collecteren van metadata uit het databasemodel van één systeem, kunnen we een template samenstellen die aangeeft welke data de gebruiker potentieel kan omzeilen.

## 2.4. TM en ARM

Voor de vraag *hoe is TM en ARM toepasbaar om omzeilde data in tekst te voorspellen* is in de literatuur gezocht op data science en organisatie onderzoek en is de vraag hieronder beantwoord.

In het organisatieonderzoek van Kobayashi et al. (2018) zijn 15 artikelen geanalyseerd op methodieken voor het toepassen van TM met aanbevelingen voor data-analyse, die aangeven dat TM gebruik maakt van de standaard data science stappen (a) preprocessing, (b) mining operations en (c) postprocessing. De preprocessing stap bestaat uit het verzamelen, schonen en transformeren van data waarbij de technieken zoals tokenization, filtering, stemming en indexing horen. Deze stappen met haar deelprocessen zijn schematisch weergegeven in figuur 1.

*Tokenization* is geschikt om teksten op te splitsen in woorden of termen die we *tokens* noemen en geeft de noodzakelijke tekstsegmentatie van de ongestructureerde tekst. Om varianten, vervoegingen en andere verschillen in vormen van hetzelfde woord naar één woord te reduceren is *stemming* beschikbaar met Natural Language Processing (NLP) en linguïstische normalisatie. Om ook de signaalinformatie te analyseren is *filtering* techniek beschikbaar, voor het detecteren van tekens (komma's, spaties, tags, opsommingstekens, formaten, signalen etc.), lidwoorden en voorzetsels, waarbij combinaties van tekst en tekens via een lijst toe te voegen zijn. Het proces *indexing* geeft de mogelijkheid om gewichtswaarde te geven aan de tokens, waarbij TF-IDF (Term Frequency en Inverse Document Frequency) het aantal keren dat het token voorkomt in de teksten relateert aan het aantal teksten dat het token bevat.




*Figuur 1. Schema van de standaard data science stappen voor TM en ARM*

Voor de stap mining operations zijn bij TM de methodieken dimensionality reduction en topic modelling geschikt voor analyse van de workarounds (Kobayashi, Mol, Berkers, Kismihok, & Den Hartog, 2018). Dimensionality reduction biedt naast filtering ook de *wrapping* techniek om subsets van gecombineerde tokens bij elkaar te voegen en *topic modelling* bouwt iteratief een kennis repository op van tokens.

Om de relaties tussen de tokens met data te analyseren zijn ARM-algoritmes beschikbaar die associatieregels genereren en daaruit kennis verwerven (Kulkarni & Kulkarni, 2016). De afsluitende postprocessing voorziet in het elimineren van toeval en het valideren van de patronen.

Uit studie blijkt dat data science standaard methodieken heeft om met TM woorden, herhalingen en signaalinformatie uit teksten te halen om daarna met ARM de relaties naar omzeilde data te leggen. De standaard data science stappen zijn toepasbaar in ons onderzoek waarbij we met tokenization, filtering en topic modelling de referentiewoorden herleiden en de signaalinformatie en frequenties als selectiecriteria gebruiken. Met ARM kunnen we de associatieregels genereren tussen de woorden en potentieel omzeilde data zodat we detectiekennis verkrijgen van deze workaround vorm.

## 2.5. ARM-Algoritmes

De vraag *welke technieken van ARM-algoritmes beschikbaar zijn voor detectie* is beantwoord met artikelen over ARM-technieken die de meest toegepaste algoritmes en technische beschikbaarheid daarvan hebben onderzocht.

ARM-algoritmes ontdekken zelf een structuur in een gegeven invoer en werken met items die met elkaar geregistreerd zijn. Deze items zijn data ontstaan uit één transacties van een gebruiker of systeem op een bepaalde plaats en periode die logisch bij elkaar horen, zoals een bestelopdracht of aankoop in een winkel. Uit deze transacties herleidt het algoritme associatieregels volgens een ALS *antecedent-item* DAN *consequentie-item* structuur. Dit houdt in dat ALS een data voorkomt DAN komt ook de andere data veel voor. Het algoritme voorspelt deze relaties met support en confidence en lift criteria wat ons inzicht geeft in de kwaliteit van relatie. De support is een indicatie dat de ALS *data* en DAN *data* tegelijkertijd voorkomen. Wanneer de ALS *data* voorkomt geeft de confidence een indicatie hoe vaak dit is met de DAN *data,* waarbij de lift inzicht geeft hoeveel de kans dat dit voorkomt hoger is dan de statistische kansberekening.

De meest toegepaste ARM-algoritmes zijn Apriori, FPGrowth en Tertius, die in gecombineerd gebruik optimale resultaten leveren (Arora, Shelza, & Rao, 2013).

*Apriori* is het bekendste algoritme en start met voor iedere data de frequentie te berekenen en komt deze frequentie boven een drempelwaarde (*support threshold*) dan analyseert het algoritme deze data. Vervolgens berekent het algoritme de frequenties van de paar-combinaties en selecteert deze wederom boven de drempelwaarde (*minimum support*). Dit proces herhaalt zich met trio's, quattro's, etc., zonder de afgevallen data mee te nemen (*pruning*).

Het *FPGrowth* algoritme start ook met het berekenen van de frequenties en bouwt een boom (*tree*) van ketens op met data in de hoogste frequentie volgorde. Het algoritme voegt data uit elke transactie in de tree toe met ophoging van de frequentie (*Growth*) waardoor er een patroon van gerelateerde data ontstaat wat we een frequent pattern tree (*FP-tree*) noemen.

Een methode die parameter gedreven is komt uit de propositielogica, waarvan de eerste-order-predicatenlogica een uitbreiding is (first-order logic) met variabelen, constanten, predicaten en functies. Het *Tertius* algoritme is hierop gebaseerd en geeft ons de meest inzichtelijke representatie van data door gebruik te maken van parameters zoals index, threshold, frequentie, ontbrekende waarde, symbolen (Arora, Shelza, & Rao, 2013).

RapidMiner en KNIME zijn de meest complete platformen om de drie algoritmes gecombineerd in te richten, waarbij RapidMiner een native FPGrowth algoritme ondersteunt en daarmee een efficiency voorkeur heeft (Jović, Brkić, & Bogunović, 2014).

De ALS-DAN-relatie detectie van een ARM-algoritme is geschikt om de relatie te onderzoeken tussen de *woorden* uit de tekst enerzijds en de *potentieel omzeilde data* anderzijds. Deze data is de input voor de ARM-algoritmes, waarbij we inzicht verkrijgen of de relatie bestaat en wat de kwaliteit is. We kunnen de drie meest toegepaste algoritmes Apriori, FPGrowth en Tertius gecombineerd toepassen in ons onderzoek om de resultaten te optimaliseren, waarbij we ons richten op het RapidMiner platform wat de breedste technologie levert.

## 2.6. Conclusie

De antwoorden op de vragen uit het literatuuronderzoek geeft ons kennis voor het ontwerp van het onderzoek en om de experimenten uit te voeren. We hebben inzicht verkregen hoe we deze workarounds in tekst detecteren en hoe we potentieel omzeilde data uit de database herleiden. We weten welke TM en ARM-algoritmes er beschikbaar zijn voor de detectie van de *ALS woord DAN omzeilde data* relaties en hoe we de voorspellende waarde optimaliseren met combinaties en criteria.

# 3. Methodologie

Voor de vraag *hoe en in welke mate TM en ARM-algoritmes toepasbaar zijn om in vrije tekst workarounds te detecteren waarmee gebruikers gestructureerde data-invoer omzeilen* gebruiken we een dataset die deze workarounds bevatten. Voor de experimenten stellen we een conceptueel en technisch ontwerp op gebruik makend van de antwoorden uit de literatuurstudie.

## 3.1. Conceptueel ontwerp

De hoofdvraag is gericht op data science technologie voor het detecteren van workarounds met als doel aanbevelingen te geven voor de praktijk en vervolgonderzoek, zodat we een exploratieve methodiek toepassen om flexibel in te spelen op voortschrijdend inzicht. We kiezen een kwantitatieve experimentele onderzoeksstrategie, omdat data science technologie een geïsoleerde statische database-omgeving vereist, met waarheidscenario's om relaties tussen variabelen met precision en recall validatie te analyseren (Saunders, Lewis, & Thornhill, 2016).

De detectie van workarounds geeft informatie over herontwerp wat aansluit op het informatiesysteem Betis van Beleste transport, dat speciaal voor deze doelstelling voor het onderwijs is ontwikkeld en qua structuur en omvang realistisch van aard is. We gebruiken de gesimuleerde database van Betis voor ons onderzoek, die de vereiste waarheidscenario's bevat in de vorm van teksten met herhalingen en signalen die een relatie hebben naar de omzeilde data. Voor de 10.000 klanten en 1,5 miljoen bestelopdrachten met vrije teksten in de database zijn voldoende workarounds opgenomen die de werkelijkheid dicht benaderen. Het informatiesysteem en de database is gedocumenteerd en door het fictieve karakter van de data zijn de wettelijke vereiste privacy en wetenschappelijke eis van publicatie gewaarborgd.

## 3.2. Technisch ontwerp

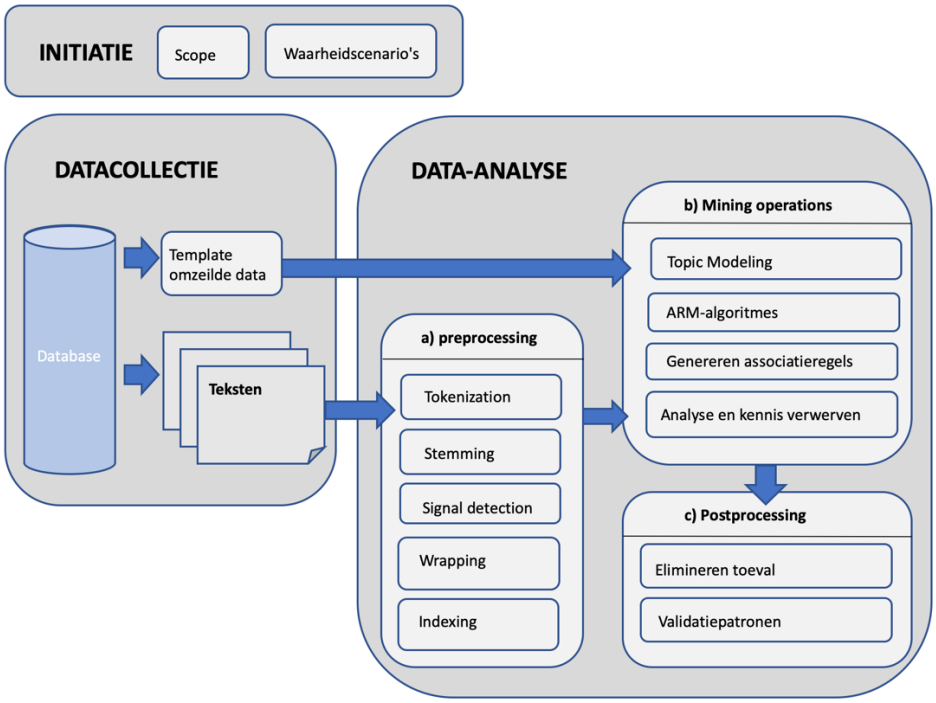
Voor het inrichten van de waarheidscenario's en evaluatie van de algoritmes gebruiken we het procedureel model met template-aanpak volgens Fürstenau en Rothe (2018) geschikt voor shadow-IT detectie met inbreng van expertkennis. De kennis van de Betis simulatieprogrammatuur is bij de initiatie nodig voor het bepalen van de waarheidscenario's en voor de datacollectie genereren we een templatetabel op basis van de kennis van het datamodel. In de data-analyse van het procedureel model zijn de standaard data science processen preprocessing, mining operations en postprocessing geïntegreerd. In figuur 3 is dit model opgesteld met de onderliggende processen en afhankelijkheden (Kobayashi, Mol, Berkers, Kismihok, & Den Hartog, 2018).

| **Teksten van de waarheidscenario's** | **Omzeilde datanaam** |
|---|---|
| gewenst op *ddmmjj* | planningsdatum |
| plannen op *ddmmjj* | |
| *ddmmjj* ophalen | |
| ophalen *ddmmjj* | |
| ophalen: *ddmmjjj* | |
| *ddmmjj* afleveren | |
| afleveren: *ddmmjj* | |
| transport *ddmmjj* | |
| *x* extra colli | colli |

*Tabel 2. Tabel met waarheidscenario-teksten en omzeilde datanaam*

Het *initiatie* proces start met het bepalen van de waarheidscenario-teksten uit de sourcecode van de simulatieprogrammatuur die zijn weergegeven in tabel 2 waarbij *ddmmjj* en *x* staat voor een omzeilde *planningsdatum* en *colli*-aantal. Uit de opdrachttabel selecteren we de opdrachten die deze teksten bevatten tezamen met de niet gevulde bijbehorende *planningsdatum* of *colli-aantal* in de structurele data. Deze opdrachten bevatten de waarheidscenario's en vormen de actuele positieven voor de precision en recall validatie.

In de *datacollectie* genereren we een template met namen, types, lengtes en de karakteristieken van de data uit het datamodel van de database. Met deze template en de opdrachtendata analyseren we of een opdracht data heeft die niet gevuld is en markeren deze als potentieel omzeilde data. Alle opdrachten met teksten en data zijn uniek door een *opdrachtnummer* en vormen de dataset van het onderzoek die we splitsen in een trainings- en testset. De trainingsset balanceren we op aanwezigheid van potentieel omzeilde data en vormt de input voor de data-analyse en de testset gebruiken we voor de validatie.




*Figuur 3. Schema van procedureel model voor Shadow-IT detectie met integratie van data science stappen*

De data-analyse start met *preprocessing* door met tokenization en stemming de woorden uit alle teksten te selecteren en met het omgekeerd toepassen van filtering de signaaltekens toe te voegen als tokens. Omdat de signaaltekens en woorden in combinatie ook een indicatie vormen van een workaround is wrapping beschikbaar om deze samen te voegen als nieuwe tokens. Om alleen de veelvoorkomende tokens te selecteren zijn filters instelbaar op frequentiewaarde door indexing.

Voor de *mining operations* zijn de tokens van de teksten en de potentieel omzeilde data de input voor het genereren van de associatieregels ALS *token* DAN *potentieel omzeilde data* met de algoritmes.

Met de waarheidscenario's bepalen we per algoritme in een confusion matrix hoeveel associatieregels waar zijn (*true positive* = TP), onwaar zijn (*false positive* = FP), hoeveel geen associatieregels hebben voor een waarheid (*false negative* = FN) en hoeveel geen

waarheid en geen associatieregels hebben (*true negative* = TN). Met de formule TP/TP+FP meten we de ratio goede matches (*precision*) en met TP/TP+FN meten we de gevoeligheid (*recall*) van het algoritme om de waarheidscenario's te detecteren. Beide geven in de formule 2 * (precision * recall) / (precision + recall) het harmonisch gemiddelde ($F_1$ *score*), waarmee we de voorspelkwaliteit tussen de algoritmes vergelijken.

Overfitting voorkomen we in de *postprocessing* door op de testset dezelfde mining operations uit te voeren, waarbij de voorspellingen uit beide confusion matrixen overeen moeten komen, waarmee we het toeval elimineren en de betrouwbaarheid borgen en de relaties valideren.

In het onderzoek doorlopen we cycli die start met initiatie en datacollectie (stap 1), waarna we door middel van observatie van de tokens uit de preprocessing (stap 2), met de kennis van de waarheidscenario's de TM-algoritmes optimaliseren. Voor de ARM-algoritmes richten we een mining proces in (stap 3) en verbeteren we de kwaliteit met parameterinstellingen en precision en recall (stap 4). Deze resultaten geven inzichten om de TM-algoritmes te verfijnen (stap 5) en om de associatieregels van de algoritmes onderling te hergebruiken (stap 6) waarna we ze valideren (stap 7). Het is een iteratief proces waarbij we na elke stap de cyclus weer opnieuw en aangepast doorlopen met voortschrijdend inzicht.

## 3.3. Data

De data is geregistreerd in een MySQL-database en bestaat uit de stamtabellen gebruiker, klant, plaats en tarief waarbij de opdrachttabel de vrije teksten bevat met relaties naar de stamgegevens. De template en dataset zijn ingericht in een MySQL-database waarbij we de vrije teksten en de potentieel omzeilde data met een Structured Query Language (SQL) selecteren, gebruik makend van foreign key en join structuren voor koppelingen naar de stamgegevens.

## 3.4. Validiteit en betrouwbaarheid

Ons onderzoek is gericht op interne validiteit en generiek toepasbaarheid van de algoritmes voor andere systemen dan het fictieve systeem Betis (Saunders, Lewis, & Thornhill, 2016). We valideren de resultaten op basis van de data science principes van trainings- en testset en gebalanceerde data. We realiseren dat het testen van de algoritmetechnieken in een andere gebruikersomgeving tot een andere resultaten zou kunnen leiden, doordat de scope van workarounds beperkt is gehouden en het maar één en fictieve casus betreft.

De data, sourcecodes, technieken, softwareversies, installaties, inrichtingen en logbestanden zijn via de Open Universiteit systemen beschikbaar en de samenstelling van het model is gedocumenteerd en reproduceerbaar met publiek beschikbare technologie en software. De Betis data voor het onderzoek is volledig en de data science componenten in ARM zijn bijgevoegd, zodat het onderzoek herhaalbaar en consistent is (Saunders, Lewis, & Thornhill, 2016).

Voor het gebruik van de Betis case in het onderzoek van de Resilience Mining afstudeerkring is toestemming verleend aan de Open Universiteit door de intellectueel eigenaar Mattic BV. De data is fictief waardoor er geen inbreuk op de privacy is en voor de in het onderzoek gebruikte software zijn legale licenties toegepast.

# 4. Resultaten

Voor het beantwoorden van de deelvragen stellen we drie experimenten op. Eerst willen we de *toepasbaarheid van TM voor het detecteren van ingevoerde tekst die data omzeilt* meten. Vervolgens onderzoeken we *hoe toepasbaar ARM is om een relatie naar omzeilde data te detecteren* en als laatste bestuderen we de *kennis van de relaties tussen ingevoerde tekst en omzeilde data*.

## 4.1. Experiment 1 – TM-algoritmes

Om antwoord te geven op de *toepasbaarheid van TM* onderzoeken we *hoe we woorden, waarden en signaalinformatie detecteren* en vervolgens *hoe we significante woorden selecteren.*

*Hoe detecteren we woorden, waarden en signaalinformatie van de workaround in de tekst?*

Als referentiewoorden en omzeilde waarden zoals *Plannen op: 011280!* in de tekst staan, dan zal TM de referentiewoorden zoals *plannen* genereren als een token. Dit woord *plannen* refereert dan bijvoorbeeld naar de planningsdatum *01-12-80* die de gebruiker in de tekst erbij heeft ingevoerd, terwijl de structurele invoer van de *planningsdatum* is overgeslagen. De gebruiker kan zowel voor als na het referentiewoord een waarde als nummer, code of datum hebben ingevoerd waarmee de ze de data omzeild.

Doordat de omzeilde waarden als *planningsdatum, klantnummer of medewerkercode* unieke waarden zijn zoals *150388*, *4520* en *wha,* zal het TM-proces ze als zelfstandige tokens detecteren. Deze tokens komen daarom weinig voor zodat we ze als irrelevant eruit filteren en we deze waarden verliezen. Als we de structuren vooraf kennen zoals een datum die door de gebruikers steeds op dezelfde manier is ingevoerd, kunnen we ze met een regular expression techniek vervangen met een sleutelwoord zoals *#ddmmjj*. Dit sleutelwoord is dan één token die aangeeft dat er een datum in de tekst voorkomt wat een indicatie kan zijn van een workaround.

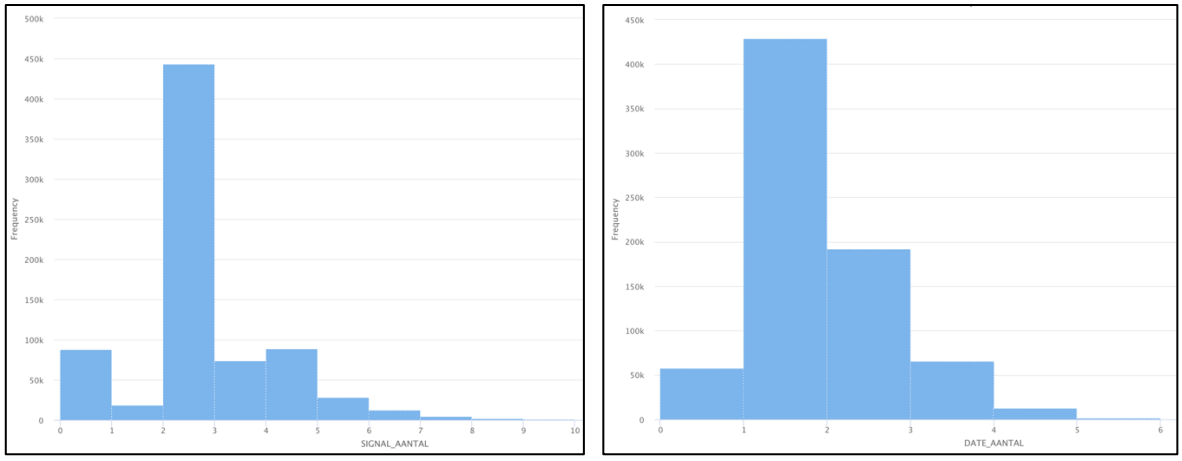
De problematiek is dat nummers, codes en datums verschillende combinaties van lengtes, nummers, letters, nullen en tussentekens kennen, die vooraf niet bekend zijn en verschillend door de gebruikers zijn ingevoerd. TM heeft hiervoor geen standaard aggregatie-, verdichting- of filteringcomponenten en verwijst naar de regular expression techniek die ontoereikend is om deze data automatisch te herkennen. Deze techniek detecteert namelijk maar één specifieke structuur die vooraf bekend moet zijn.

Als we in staat zijn om de waarden wel te herleiden ontstaat er een nieuwe uitdaging dat deze waarden ook voor andere doeleinden dan een workaround zijn gebruikt, zodat ze zelfstandig geen significant onderscheid geven voor detectie van een workaround.

Dezelfde problemen ondervinden we met de signaaltekens die signaalinformatie geven over een workaround. Gebruikers maken zelf structuur aan en kunnen signaaltekens zoals *:,!-/;* tegen het woord aanplakken zoals bij *Spoed!!* of *-klant* en met tokenizing filteren we deze tekens eruit om het woord te herleiden. We verliezen daardoor de signaaltekens uit de tekst die juist een indicatie kunnen zijn van een workaround. Ook transformeert het proces de woorden naar kleine letters, zodat het woord *Spoed* en *spoed* hetzelfde token krijgt, maar de informatie dat het woord met één of meerdere hoofdletter is geschreven verdwijnt.

Om bovenstaande signaalinformatie niet te verliezen is een algoritme nodig om de combinaties van referentiewoorden en signaalinformatie te classificeren. Hiervoor biedt TM geen oplossing en verwijst naar de regular expression techniek, waarbij we bijvoorbeeld met de regular expression "- */w*" iedere combinatie van *streepje* gevolgd door *spatie* en een *woord* kunnen detecteren. Op deze wijze zouden we combinaties van signaaltekens of hoofdletters kunnen vervangen voor een sleutelwoord zoals *#signaalinformatie* dat in de tekst aangeeft dat er extra signaalinformatie is. Het probleem is wederom dat deze techniek maar één specifieke combinatie kan herkennen, terwijl het aantal mogelijke combinaties tussen woorden, signaaltekens en hoofdletters groot zijn en vooraf niet bekend.

Een alternatief is om met dezelfde techniek alle niet signaaltekens eruit te filteren, door de regular expression *[ -!"#$%&'()*+,./:;<=>?@\[\\\]_`{|}~]* te gebruiken, waardoor alleen de signaaltekens over blijven. Met de aantallen en/of soort signaaltekens in de tekst kunnen we per tekst analyseren of deze signaalinformatie bevat, alleen verliezen we de relatie met de referentiewoorden en waarden. Als gebruikers de signaaltekens frequent gebruiken voor andere doeleinden dan het structureren van een workaround geeft deze informatie ook geen significant indicatie van een workaround.




*Figuur 4. Grafieken van frequentie aantal signaaltekens (links) en datums (rechts) in één tekst*

In ons onderzoek zijn datums consequent met alleen dag, maand en jaar getallen ingevoerd zonder streepjes, zodat we deze transformeren met een regular expression "999999" voor een volledige datum en "9999" als het jaar is weggelaten. In figuur 4 zien we rechts een grafiek met de frequenties van de aantallen keren dat een datum in één tekst voorkomt. Hieruit blijkt dat de datums in vrijwel alle teksten voorkomen en veelal meer dan 1 keer, zodat ze als zelfstandige waarde in de tekst niet onderscheidend zijn voor de workarounds. In de grafiek aan de linkerkant zien we de frequenties van de aantallen signaaltekens in één tekst met dezelfde kenmerken zodat deze ook niet onderscheidend zijn.

Met TM kunnen we adequaat referentiewoorden uit tekst detecteren alleen is het minder geschikt om waarden en signaalinformatie-structuren te herleiden die ook een indicatie zijn van een workaround. Omdat we vooraf geen kennis hebben van de structuren met signaaltekens, nummers, codes en datums in de teksten missen we een standaard algoritme die deze structuren intelligent herkent. Het bestuderen van alle tokens op deze structuren, waarna we ze vervangen met een regular expression techniek is een tijdsintensief en semi-automatische proces.

*Hoe selecteren we significante woorden uit de teksten als indicatie van een workaround?*

Het TM-proces genereert tokens van alle woorden in de tekst en de hoeveelheid is te tijdsintensief voor de ARM-algoritmes om te verwerken. De *stemming* en lengte beperking is niet afdoende om deze aantallen te beperken, zodat een *pruning* technologie nodig is om te filteren op de aantallen dat de tokens voorkomen in alle teksten. TM biedt daarvoor een ondergrens en bovengrens frequentiefilter aan om de significante woorden te selecteren.

De bovengrens is bedoeld om vaak gebruikte irrelevante woorden zoals *heeft* of *omdat* in proza weg te filteren uit de documenten. Omdat vrije tekst in informatiesystemen bedoeld zijn om *extra* korte informatie weer te geven, die niet in de structurele invoer getikt kan worden, bevat ze geen proza en vervalt het doel van de bovengrens. Een kenmerk van workarounds is dat ze juist frequent voorkomen, doordat ze herhalend en routinematig zijn toegepast door de gebruikers zodat we de bovengrens niet instellen.

De ondergrensfiltering is wel een geschikte methode om de niet significantie woorden te verwijderen zoals weinig gebruikte woorden en fouten van gebruikers. Deze filter reduceert het aantal tokens substantieel ten gunste van de verwerkingstijd van ARM. Met de ondergrens filteren we sporadisch voorkomende woorden zoals *meebetalingsachterstand* eruit, maar ook verkeerd gespelde woorden die zonder spatie of onderscheidingsteken aan elkaar zijn gekoppeld.

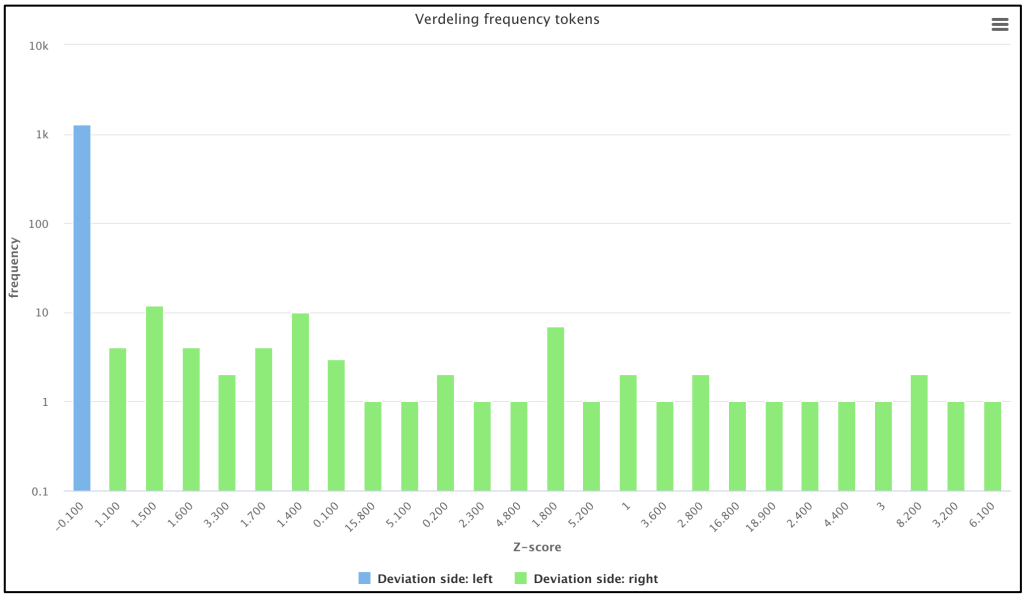
Om de ondergrens te bepalen is de *indexing* techniek TF-IDF gangbaar. Ze maakt gebruik van de aantallen teksten dat een token voorkomt wat de *document*frequentie heet in combinatie met de aantallen keren dat een token in alle teksten voorkomt wat de *totaal*frequentie is. Hiermee bepaalt het algoritme de belangrijkheid van het woord in de corpus tekst, maar omdat we te maken hebben met korte teksten en geen proza is deze indexing niet effectief. De beide frequenties komen in teksten vrijwel overeen, omdat elk token meestal maar één keer voorkomt in een tekst, zodat we de *indexing* techniek BTO (Binary Term Occurances) toepassen die uitgaat van het *wel* of *niet* voorkomen van een token in de tekst.

De niet significante woorden vormen een grote groep tokens met lage frequenties en de significante woorden zijn een beperkt aantal tokens met hogere frequenties. De tokenfrequenties hebben een scheefheid verdeling naar rechts ten opzichte van het gemiddelde die we kunnen gebruiken om significantie tokens te selecteren. Het gemiddelde van de frequenties vormt de ondergrens van *pruning* waarmee we de vervuiling scheiden en waardoor significante woorden over blijven die de woorden van de workarounds bevatten.

In de grafiek van figuur 5 zien we deze verdeling van de frequentie van de tokens in een Z-score weergegeven, waarbij de scheefheid naar rechts zichtbaar is en is afgezwakt door een logaritmische y-as. In de tabel aan de rechterkant zijn een deel van de tokens weergegeven met de frequenties waarbij het gemiddelde ligt tussen 332 en 2239, wat we hebben aangegeven met een groene streep.

De referentiewoorden van de workarounds zijn tokens zoals *ophal* en *aflever* en staan bovenaan in de lijst met hoge frequenties en een woord zoals *zoekge;raakt*, waarbij de gebruiker een extra *";"* per ongeluk heeft ingetikt is opgesplitst in twee tokens *zoekge* en *raakt* en komt als vervuiling maar één keer voor in de teksten.

De gemiddelde frequentie die een Z-score van 0 heeft is geschikt om als ondergrens te dienen voor de *pruning* waardoor we de vervuiling eruit filteren. We realiseren hiermee een reductie van 1363 tokens naar 67 tokens waardoor er 5% van de tokens als significant overblijft.




| word | frequency |
|---|---|
| nieuw | 9859 |
| aannem | 9838 |
| credit | 9386 |
| crediter | 9319 |
| korting | 9234 |
| ingetrok | 2381 |
| annul | 2322 |
| geannul... | 2291 |
| afgezegd | 2269 |
| annuler | 2239 |
| pnoniet | 332 |
| tbiniet | 324 |
| hmeniet | 310 |
| oniniet | 307 |
| pbaniet | 305 |
| areniet | 302 |
| lnaniet | 294 |

*Figuur 5. Grafiek van de verdeling frequentie van de tokens (links) en een tabel met een deel van de tokens met de frequenties (rechts)*

Met TM pruning techniek en scheiding op gemiddelde kunnen we de verzameling woorden uit de teksten substantieel reduceren met behoud van de referentiewoorden van de workarounds, zodat we een significante verzameling tokens kunnen aanbieden voor de ARM-algoritmes.

## 4.2. Experiment 2 – ARM-algoritmes

Om antwoord te geven op de vraag *hoe toepasbaar is ARM om een relatie naar omzeilde data te detecteren* dienen we te onderzoeken *hoe we bepalen welke data-invoer de gebruiker potentieel heeft omzeild.* De tokens uit het TM-proces samen met deze potentieel omzeilde data geven we als input aan de ARM-algoritmes met de vraag *welke relaties de Apriori, FPGrowth en Tertius algoritmes genereren.* Tot slot bepalen we *wat de voorspelkwaliteit is van de relaties.*

### *Hoe bepalen we welke data-invoer de gebruiker potentieel heeft omzeild?*

Een gebruiker omzeilt data door invoer over te slaan en de informatie in de tekst op te nemen, zodat de verzameling overgeslagen data potentieel omzeilde data kan bevatten. Deze overgeslagen data heeft *geen* inhoud, met als uitzondering als de data vooraf is ingevuld door het systeem met een *standaard*waarde zoals ‘Nee’ of ‘Onbekend’. De gebruiker kan deze data ook overslaan, maar het systeem slaat dan *wel* een inhoud op. Er is ook invoer die de gebruiker *niet* kan overslaan, omdat het een verplichte invoer is zoals de naam van de klant.

De vraag welke data de gebruiker kan overslaan, is te beantwoorden met het doorlopen van de invoerschermen van het systeem. De inrichting van deze schermen is veelal gericht op een bepaalde activiteit zoals het registreren van een bestelopdracht en kan bestaan uit meerdere invoerschermen. De database beschikt ook over deze informatie en is automatisch te herleiden in tegenstelling tot het inventariseren van de schermen.

In de tabellen van een database is de data die de gebruiker kan invoeren opgeslagen in rijen en kolommen. In de metadata van de kolommen is vastgelegd dat data overgeslagen mag worden door de gebruiker, met de inrichting dat een kolom lege data mag bevatten of dat er een standaardwaarde is.

Welke in te voeren structurele data bij een vrije tekst-invoer hoort is ook bekend via deze inrichting. Een tekst is namelijk als een kolom opgenomen in een tabel en de andere kolommen van de tabel geven de bijbehorende structureel in te voeren data weer. Zo'n tabel kan relaties hebben naar andere tabellen, die ook de bijbehorende in te voeren data bevatten.

Door deze kolommen automatisch te selecteren uit het datamodel van de database, herleiden we de data-invoer van de gebruiker behorende bij de vrije tekst. Aan de hand van deze metadata vormen we een template *structurele data-invoer,* die aangeeft welke data leeg mogen zijn en of ze een standaardwaarde hebben. Met deze kennis kunnen we de inhoud van de kolommen onderzoeken of ze daadwerkelijk kenmerken bevatten dat de gebruiker deze heeft overgeslagen en dus potentieel omzeild kunnen zijn. Door de potentieel omzeilde data te inventariseren krijgen we inzicht welke data een gebruiker veel overslaat, wat een indicatie is dat de gebruiker deze data in tekst kan hebben omzeild.

| TABLE_NAME | COLUMN_NAME | IS_NULLABLE | COLUMN_TYPE |
|---|---|---|---|
| klant | NAAM | NO | varchar(80) |
| klant | CP | YES | varchar(80) |
| klant | STRAAT | NO | varchar(80) |
| klant | PC | NO | char(6) |
| klant | HUISNR | NO | varchar(20) |
| klant | PLAATS | NO | varchar(60) |
| klant | TEL | YES | varchar(40) |
| klant | BLOK | NO | char(1) |
| opdracht | DOPDR | NO | date |
| opdracht | COLLI | YES | int(11) |
| opdracht | KG | YES | decimal(5,2) |
| opdracht | STRAAT | NO | varchar(80) |
| opdracht | PC | NO | char(6) |
| opdracht | HUISNR | NO | varchar(20) |
| opdracht | PLAATS | NO | varchar(60) |
| opdracht | DPLAN | YES | date |
| opdracht | DTRANS | YES | date |
| opdracht | BONBIN | YES | char(1) |
| opdracht | BEDRAG | YES | decimal(6,2) |

*Tabel 6 MySQL-templatetabel met structurele data-invoer van de opdracht*

Voorwaarde voor het automatisch genereren van de template is dat de relationele verwijzingen naar de tabellen consistent is toegepast, aangezien inconsistentie tussen tabellen *valse* lege data kan opleveren bij het koppelen van de tabellen. Een ander probleem is dat dat kolommen beperkt zijn ingericht, zodat we systeemdocumentatie moeten raadplegen of kennis van een applicatie noodzakelijk is.

De metadata over de kolommen is in MySQL beschikbaar vanuit de tabel COLUMN in de metadatabase *information_scheme*. Een kopie van deze tabel geeft ons de metadata voor de template die we in tabel 6 hebben opgenomen en waarin de structurele data-invoer per regel

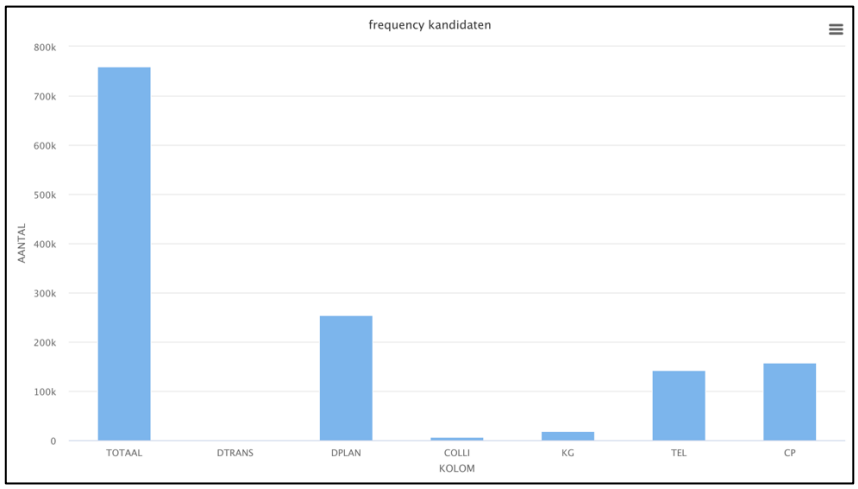
is weergegeven. Deze regels zijn afkomstig van de metadata van de opdrachttabel en omdat deze een relatie heeft naar de klanttabel zijn deze ook als data-invoer meegenomen.

| NR | COLLI | KG | DPLAN | DTRANS | CP | TEL |
|---|---|---|---|---|---|---|
| 1626 | false | false | true | false | false | false |
| 1633 | false | false | false | false | false | false |
| 1634 | false | false | true | false | false | false |
| 1637 | false | false | false | false | false | false |
| 1641 | false | false | false | false | false | false |
| 1653 | false | false | false | false | false | false |
| 1657 | false | false | false | false | false | false |
| 1667 | false | false | true | false | false | false |
| 1695 | false | false | false | false | true | false |
| 1700 | false | false | false | false | false | false |
| 1710 | false | false | false | false | true | false |
| 1712 | false | false | false | false | false | true |
| 1722 | false | false | false | false | false | false |
| 1727 | false | false | false | false | true | false |

*Tabel 7. De RapidMiner tabel met potentieel omzeilde data per opdracht*

We constateren een inconsistentie omdat de opdrachttabel geen betrouwbare verwijzing heeft naar de klant- en gebruikerstabel wat we eerst dienen te herstellen. Zo ontdekken we dat er klantnummers bestaan in opdrachten die niet in de klanttabel voorkomen. Bij het koppelen van de tabellen met een SQL join techniek levert dit *valse* lege data op die we eruit filteren. Ook is er één onvolledige inrichting bij de kolom BONBIN, waarbij het onbekend is of deze standaard op *'N'* staat. Voor de resterende 6 kolommen is het duidelijk dat de instelling IS_NULLABLE in de template bij *'YES'* aangeeft dat ze leeg mogen zijn en dat de gebruiker deze kan overslaan.

Deze 6 kolommen in de tabellen hebben we met de SQL-taal onderzocht of ze ook overgeslagen inhoud bevatten die we markeren als potentieel omzeilde data. In de RapidMiner tabel 7 zijn de opdrachten in rijen zichtbaar met een *true* of *false* markering voor potentieel omzeilde data. Zo heeft het groen gearceerde opdrachtnummer 1626 een *true* potentieel omzeilde data in de kolom DPLAN, wat inhoudt dat de gebruiker de *planningsdatum* heeft overgeslagen en daarmee potentieel omzeild kan zijn.




| KOLOM | AANTAL | PERCENTAGE |
|---|---|---|
| TOTAAL | 758663 | 100 |
| DTRANS | 1746 | 0 |
| DPLAN | 253730 | 33 |
| COLLI | 7564 | 1 |
| KG | 18458 | 2 |
| TEL | 143181 | 18 |
| CP | 157579 | 20 |

*Figuur 8. Grafiek van aantallen potentieel omzeilde data per kolom (links) met tabel kolommen, aantallen en percentage (rechts)*

In figuur 8 geven we de grafiek (links) en tabel (rechts) weer met de aantallen en percentage potentieel omzeilde data per kolom en het totaal aan opdrachten, waarbij zichtbaar is dat de gebruiker de *planningsdatum* in 33%, *telefoon* in 18% en *contactpersoon* in 20% van de gevallen over slaat en dit bij *transportdatum*, *colli* en *kilogram* sporadisch doet.

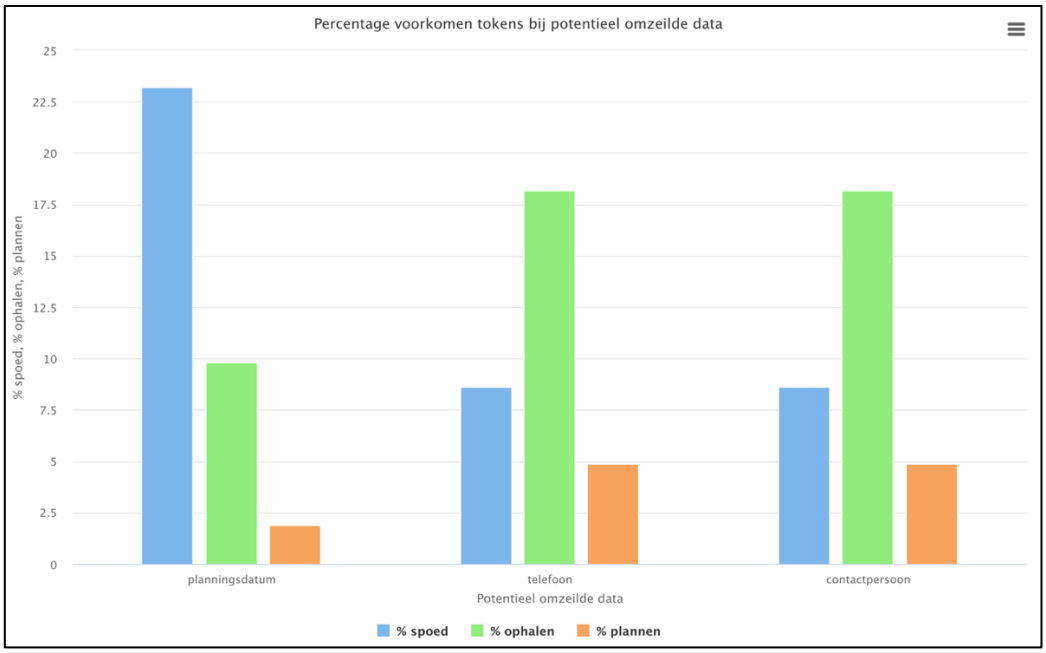
Met de metadata uit de database en SQL-techniek herleiden we de data die de gebruiker overslaat automatisch, wat inzicht geeft in de potentieel omzeilde data die we kunnen aanbieden voor de ARM-algoritmes. Als de inrichting van de database niet consistent en volledig is, zal systeemdocumentatie of kennis van de applicaties noodzakelijk zijn en het proces semi-automatisch maken.

### *Welke relaties genereren de Apriori, FPGrowth en Tertius algoritmes?*

De Apriori en FPGrowth algoritmes genereren bruikbare ALS *token* DAN *potentieel omzeilde data* relaties en beide algoritmes vinden dezelfde relaties die workarounds bevatten. Ze zijn beide geschikt alleen geeft FPGrowth van RapidMiner beter analyseerbare resultaten zodat deze voorkeur heeft. Het Tertius algoritme is ongeschikt voor workarounds omdat het ook negatieve relaties met ALS *geen token* DAN *geen potentieel omzeilde data* levert, wat betrouwbare relaties zijn en daardoor geen positieve relaties geeft die de workarounds bevatten.

### Gehele datasetanalyse

Als we de gehele dataset analyseren met de ARM-algoritmes, genereren zij betrouwbare relaties tussen tokens onderling met een ALS *token* DAN *token* regel. Deze regels geven kennis van gecombineerd woordgebruik zoals het woord *bezorgen* dat veel voorkomt met *klant,* maar deze combinaties zijn geen workaround en vallen buiten onze scope van het onderzoek. De algoritmes vinden ALS *token* DAN *potentieel omzeilde data* relaties, maar deze zijn onbetrouwbaar of komen niet overeen met de referentiewoorden van de workarounds.




*Figuur 9. Grafiek van percentage voorkomen tokens bij potentieel omzeilde data*

De referentiewoorden van de workarounds zijn tokens zoals *ophal* en *aflever* die voor de algoritmes geen betrouwbare relaties vormen, omdat deze woorden ook gebruikt zijn om andere informatie vast te leggen. Ze zijn dus niet exclusief genoeg gerelateerd aan de potentieel omzeilde data.

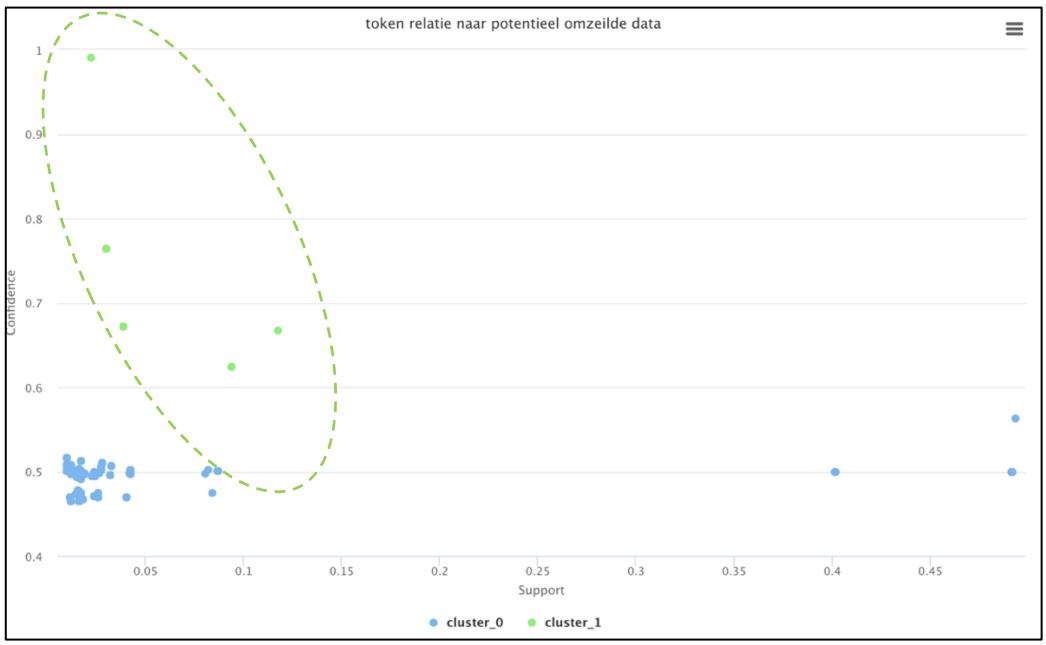
In de gehele dataset komen ALS *token* DAN *potentieel omzeilde data* relaties voor tussen de 1% en 4% en zijn niet meer dan 40% betrouwbaar. De enige betrouwbare relatie is ALS *spoed* DAN *potentieel omzeilde planningsdatum* met een betrouwbaarheid van 91%. Dat er met het woord *spoed* in tekst de *planningsdatum* is overgeslagen geeft interessante kennis, maar het is geen workaround die wij zoeken omdat er geen planningsdatum in de tekst is gezet.

In figuur 9 zien we een grafiek met de percentages dat de woorden *spoed*, *ophalen* en *plannen* voorkomen in drie deelverzamelingen potentiele omzeilde data. Zichtbaar is dat de referentiewoorden *ophalen* en *plannen* frequent voorkomen bij potentieel omzeilde *telefoon* en *contactpersoon* data en daardoor niet exclusief zijn bij potentieel omzeilde *planningsdatum* data. Zichtbaar is dat het woord *spoed* wel vaker gebruikt is bij deze potentieel omzeilde *planningsdatum* waardoor dit woord exclusiever is en als een betrouwbare relatie is gevonden door de algoritmes.

Referentiewoorden kunnen ook gebruikt worden voor tekst zonder dat het een workaround is, waardoor ze over de gehele dataset voorkomen bij verschillende potentieel omzeilde data. De algoritmes classificeren ze daardoor als onbetrouwbare relaties zodat analyse van de gehele dataset niet effectief is.

### Deelverzameling analyse

Om de betrouwbare relaties met referentiewoorden in de dataset te vinden, kiezen we voor een analyse per gebalanceerde deelverzameling potentieel omzeilde data. Omdat de relaties binnen potentieel omzeilde data anders zijn dan daarbuiten vindt het FPGrowth algoritme nu wel betrouwbare relaties waar ook de workarounds in voorkomen omdat de tokens de referentiewoorden bevatten.




*Figuur 10. Grafiek van confidence en support van de relaties naar potentieel omzeilde data met clustering op confidence*

Om de *significante* relaties te selecteren is de betrouwbaarheid de onderscheidende factor. Een kleine groep relaties met een hoge betrouwbaarheid zijn duidelijk onderscheidend met een grote groep relaties met een lage betrouwbaarheid. Deze groepen hebben centrale middelpunten waardoor we ze met het clustering algoritme K-Means detecteren. Het cluster met hoge kwaliteit relaties bevatten de referentiewoorden en de scheiding met de lage kwaliteit vormt het selectiecriteria.

We hebben deelverzamelingen gemaakt van de 6 potentieel omzeilde data en deze gebalanceerd door ze te verdubbelen met opdrachten waarbij de data niet potentieel omzeild is. Alleen de deelverzameling *planningsdatum*, *telefoon* en *contactpersoon* geven ALS *tokens* DAN *potentieel omzeilde data* relaties. In figuur 10 is een grafiek weergegeven van deze relaties waarbij we de betrouwbaarheid (*confidence*) afzetten tegen de frequentie (*support*) van de relaties.

Op basis van een K-Means algoritme zijn er 2 clusters berekend op de betrouwbaarheid, weergegeven met de lage (blauw) en hoge (groen) betrouwbaarheid. Het cluster met een betrouwbaarheid van 60% of hoger die we groen omcirkelen, geven de significante relaties weer met de referentiewoorden.

Deze 5 significantie relaties geven de kennis dat de 5 tokens *gewenst, plann, transport, ophal* en *aflever* voorkomen met een *potentieel omzeilde planningsdatum*, waarbij ze 8 van de 9 referentiewoorden van de waarheidscenario's voorspellen. In tabel 11 is de betrouwbaarheid, support en lift van deze relaties weergegeven, waarbij het FPGrowth algoritme van RapidMiner de benaming *premises* gebruikt voor ALS en *conclusion* voor DAN van de relatie. De betrouwbaarheid ligt tussen de 62% en 99% en de kans dat de relatie voorkomt is tussen 1,2 en 1,9 keer hoger dan de statistische kans berekening zodat we toeval kunnen uitsluiten.

| Premises | Conclusion | Confide... ↓ | Support | Lift |
|---|---|---|---|---|
| gewenst | DPLAN | 0.991 | 0.022 | 1.982 |
| plann | DPLAN | 0.764 | 0.030 | 1.528 |
| transport | DPLAN | 0.672 | 0.039 | 1.344 |
| ophal | DPLAN | 0.667 | 0.118 | 1.334 |
| aflever | DPLAN | 0.624 | 0.094 | 1.248 |

*Tabel 11. RapidMiner tabel met 5 relaties ALS token DAN potentieel omzeilde planningsdatum gedetecteerd in de deelverzamelingen*

De ARM-algoritmes FPGrowth en Apriori zijn geschikt voor detectie van ALS *token* DAN *potentieel omzeilde data* relaties in tegenstelling tot het Tertius algoritme. Voor het vinden van relaties die overeenkomen met referentiewoorden van de workarounds, is een analyse per omzeilde dataverzameling nodig en kunnen we met clustering op hoge en lage betrouwbaarheid de voorspellende relaties detecteren.

### *Wat de voorspelkwaliteit is van de relaties?*

De gevonden ALS *token* DAN *potentieel omzeilde data* relaties zijn volledig voorspellend voor de referentiewoorden die worden gebruikt bij de workarounds. In feite geven de relaties kennis over de ALS *referentie woord* DAN *omzeilde data* die voorkomt in de workaround teksten. Deze relaties voorspellen weliswaar de meeste waarheidscenario's maar ook veel situaties dat het geen workaround betreft.

| Token | Referentiewoord | Teksten van de waarheidscenario's | | |
|---|---|---|---|---|
| gewenst | gewenst | *gewenst op ddmmjj* | | |
| plann | plannen | *plannen op ddmmjj* | | |
| ophal | ophalen | *ddmmjj ophalen* | *ophalen ddmmjj* | *ophalen: ddmmjj* |
| aflever | afleveren | *ddmmjj afleveren* | *afleveren: ddmmjj* | |
| transport | transport | *transport ddmmjj* | | |

*Tabel 12. Tabel met relaties tussen tokens, referentiewoorden en workaround teksten*

Tabel 12 geeft inzicht hoe de gevonden tokens overeen komen met de referentiewoorden van 8 van de 9 waarheidscenario-teksten, waarbij *ddmmjj* staat voor een ingevoerde *planningsdatum* in de tekst die omzeild is bij de structurele datum-invoer.

In de confusion matrix van tabel 13 is zichtbaar dat er geen false negatives (FN) zijn waardoor de recall 100% is. De false positives zijn duidelijk zeer groot wat komt omdat de referentiewoorden ook in teksten voorkomen zoals *transport verzekeren*, *klant is afleveradres* en *niet ophalen maar brengen* wat geen workaround teksten zijn en waardoor de precision maar 3% is.

| VOORSPEL | WAAR | TP | FP | FN | TN |
|---|---|---|---|---|---|
| false | false | 0 | 0 | 0 | 120729 |
| true | false | 0 | 55262 | 0 | 0 |
| true | true | 1523 | 0 | 0 | 0 |

*Tabel 13. RapidMiner confusion matrix - ALS gewenst, plann, transport, ophal of aflever DAN omzeilde planningsdatum*

De door ARM gegenereerde relaties voorspellen de meeste referentiewoorden die gebruikt zijn voor een workaround, maar omdat deze woorden ook door de gebruiker worden ingevoerd voor een niet workaround voorspellen ze de individuele workarounds slecht.

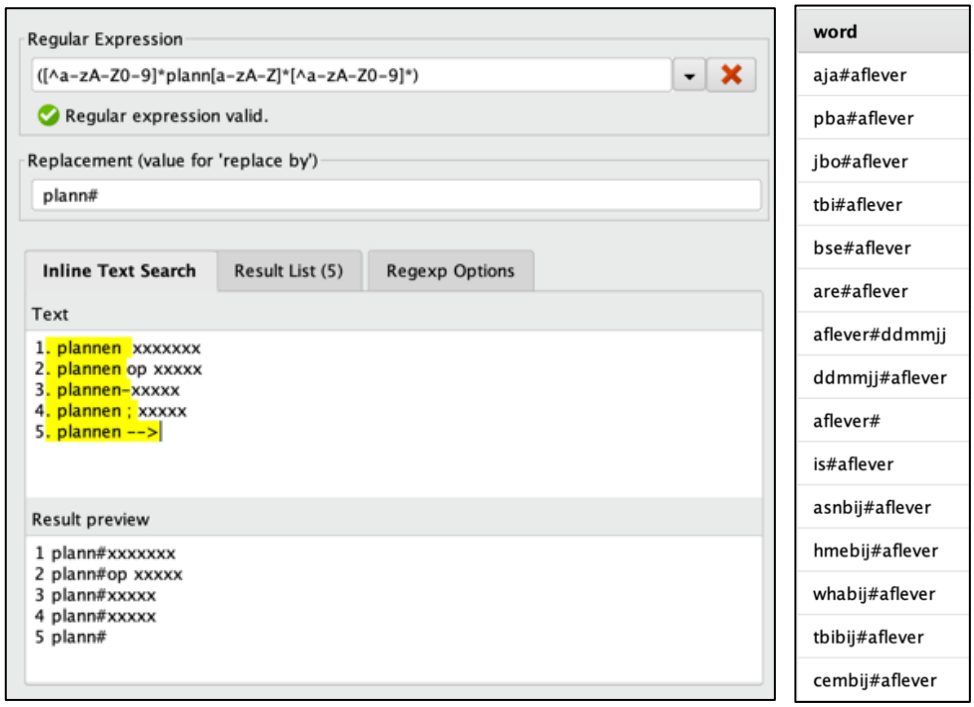
## 4.3. Experiment 3 – Kennis workaround

Om antwoord te geven op de vraag *welke kennis de relaties geven tussen ingevoerde tekst en omzeilde data* hebben we inzicht te verkrijgen in *de ingevoerde tekst bij de relatie* waarna we onderzoeken *welke ingevoerde tekst significant is.* Tot slot bepalen we *wat de voorspelkwaliteit is van de ingevoerde tekst.*

### *Welke kennis van de ingevoerde tekst bij de relatie kunnen we herleiden?*

Doordat we in staat zijn om de referentiewoorden te detecteren hebben we een betrouwbaar *anker* in de tekst om de woorden en nummers voorafgaand (*pre*) of nakomend (*post*) bij de referentiewoorden te herleiden, zodat we kennis verkrijgen van de workaround tekst. We kunnen de pre of post woorden of nummers met de eerder beschreven regular expression techniek herleiden. Door deze methode plakken we unieke waarden van *datums*, *nummers* of *codes* tegen het referentiewoord aan zodat we ze herkennen en aggregatie van de waarde mogelijk is.

Een handicap is dat we ook tussenvoegsels zoals *op, in, bij* er aan plakken, die veelal daarvoor of erna nog een waarde bevat die we niet detecteren. Een ander probleem is dat we ook samenstellingen zoals *planningsmethode* of *afleveradres* selecteren die het token van het referentiewoord bevatten. Doordat deze samenstellingen ook als unieke tokens voorkomen kunnen we ze automatisch detecteren en eruit filteren.



*Figuur 14. De ingestelde regular expression in RapidMiner (links) en de combinatietokens van afever (rechts)*

Figuur 14 geeft aan de linkerkant een voorbeeld van hoe we met het token *plann* de voluit geschreven referentiewoorden zoals *plannen* inclusief de tekens erachter kunnen detecteren. Zo veranderen we de tekst *plannen* met het teken *";"* naar één vervangend combinatietoken *plann#,* waarachter direct het volgende woord of nummer komt. Het geel gearceerde voorbeeld laat een test zien dat de opvolgende dummy tekst *xxxxx* direct achter *plann#* komt en één combinatietoken vormt zoals *plann#xxxxx.* We kunnen hetzelfde doen met de woorden of getallen die voor het woord *plannen* komen zodat *xxxxx#plann* ontstaat.

De combinatietokens met de aangeplakte woorden van het token *aflever* zijn weergegeven in figuur 14 aan de rechterkant. Zichtbaar zijn de unieke waarden zoals *aja, pba*, of *jbo* die voor het token komen en dit zijn medewerkerscodes die we aggregeren met het sleutelwoord *medewerkercode*. Voor de datum is deze aggregatie met het sleutelwoord *ddmmjj* al geschied zodat *ddmmjj#aflever* alle datums geeft voorafgaand aan het woord *aflever*.

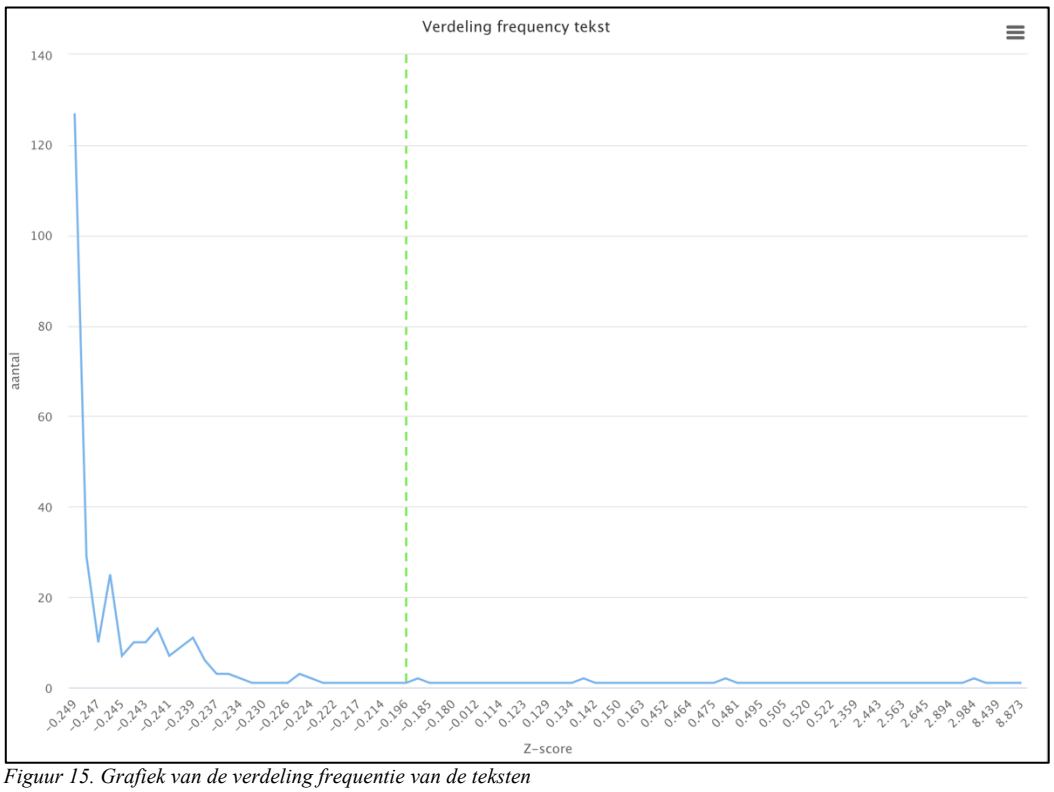
Doordat we met de ARM-algoritmes de *referentiewoorden* detecteren hebben we een middel om de tekst rond deze woorden te vinden met de regular expression techniek, waardoor we kennis krijgen over de workaround tekst. Zo detecteren we of er omzeilde data zoals een *datum*, *nummer* of *code* erbij zijn ingevoerd, waarbij we unieke waarden aggregeren en omzeilde data in de tekst herkennen. Deze teksten zijn een sterkere indicatie van de workaround dan alleen het referentiewoord.

### *Welke kennis van de ingevoerde tekst bij de relatie is significant?*

De aantallen gecombineerde woorden rond het referentiewoord zijn te groot om kennis te verkrijgen en vraagt om reductie op significantie door een semi-automatisch proces. De gevonden waarden bij de referentiewoorden bevatten geaggregeerde *datums*, *nummers* of *codes*, die geen relatie hebben met deze omzeilde data en niet significant zijn. Zo heeft de *medewerkercode* bij de omzeilde *planningsdatum* geen waarde zodat we deze teksten eruit filteren.

Ook zijn er in de teksten vervuilingen zoals *wu#aflever* wat niet significante combinaties van woorden zijn en een lage frequentie hebben. De verdeling heeft dezelfde scheefheid naar rechts als de enkelvoudige tokens, maar door de aggregatie van de waarden is de scheiding voor *pruning* verschoven onder het gemiddelde. Dit houdt in dat we de *pruning* voorafgaand aan de aggregatie uitvoeren wat vraagt om maatwerkprogrammering.

De kleine lijst met significante teksten die over blijft is taalkundig te analyseren, door te controleren of de waarde zoals datum is ingevoerd om bijvoorbeeld de *planningsdatum* invoer te omzeilen.



*Figuur 15. Grafiek van de verdeling frequentie van de teksten*

In de grafiek in figuur 15 zien we de verdeling van de frequenties van teksten in Z-score met de scheefheid verdeling naar rechts. De vervuilingsgrens is aangegeven met een groene lijn en ligt onder het gemiddelde, doordat de aggregatie van *datums* het gemiddelde heeft verhoogd. Na *pruning* en het verwijderen van de teksten met *medewerkerscodes,* reduceren we de lijst naar 20 significant teksten waarbij het koppelteken # is verwijderd.

Uit de 20 teksten die in tabel 16 links zijn weergegeven analyseren we dat alle combinaties met *ddmmjj* refereren naar een omzeilde *planningsdatum*. Bij *gewenst op* en *plann op* is het duidelijk dat het woord *op* gebruikt is om te verwijzen naar een datum die daarna komt. Uiteindelijk resulteert deze taalkundige analyse in een beperkte lijst van 8 workaround teksten zichtbaar in tabel 16 aan de rechterkant.

| word | in docu... |
|---|---|
| ophal maar | 10428 |
| aflever bij | 10231 |
| niet ophal | 9932 |
| aflever identificatie | 3761 |
| op aflever | 3696 |
| aflever adres | 3696 |
| transport deken | 3691 |
| achterom ophal | 3459 |
| transport verzekeren | 3309 |
| aflever niet | 3274 |

| word | in docu... |
|---|---|
| ophal niet | 3209 |
| opnieuw plann | 3072 |
| ophal ddmmjj | 816 |
| plann ddmmjj | 272 |
| aflever ddmmjj | 103 |
| ddmmjj aflever | 80 |
| gewenst op | 79 |
| ddmmjj ophal | 74 |
| plann op | 69 |
| transport ddmmjj | 68 |

| word | in docume... |
|---|---|
| ophal ddmmjj | 816 |
| plann ddmmjj | 272 |
| aflever ddmmjj | 103 |
| ddmmjj aflever | 80 |
| gewenst op | 79 |
| ddmmjj ophal | 74 |
| plann op | 69 |
| transport ddmmjj | 68 |

*Tabel 16. De significante teksten (links) en de taalkundig gefilterde workaround teksten (rechts) uit RapidMiner*

De lijst met gecombineerde woorden rond het referentiewoord is adequaat taalkundig te analyseren op significante aspecten zoals tussenvoegsels en waarden die verwijzen naar de omzeilde data en daarmee een workaround zijn. De lijst is gereduceerd door het verwijderen van vervuiling in combinatie met de aggregatie van waarden wat om maatwerk programmeren vraagt.

### *Wat is de voorspelkwaliteit is van de ingevoerde tekst bij de relatie?*

De gevonden significante teksten op basis van de referentiewoorden voorspellen de workarounds met een zeer hoge betrouwbaarheid.

In tabel 17 is de confusion matrix weergegeven van de voorspelling van alle significante teksten zoals *plan op* en *ddmmjj aflever.* Ze geven gezamenlijk een precision, recall en $F_1$ score van bijna 100%, wat een hoge betrouwbaarheid is. Er zijn een paar false negative (FN) omdat de gebruikers tussen de medewerkerscode en datum geen spatie hebben ingetikt, zodat bijvoorbeeld de token *bseddmmjj ophalen* ontstaat die eruit is gefilterd met een te lage frequentie. De enkele false positives (FP) zijn situaties waarbij de gebruiker de datum niet correct heeft ingevoerd maar door het woord *op* wel zijn voorspeld als workaround.

| VOORSPEL | WAAR | TP | FP | FN | TN |
|---|---|---|---|---|---|
| false | false | 0 | 0 | 0 | 175988 |
| false | true | 0 | 0 | 5 | 0 |
| true | false | 0 | 3 | 0 | 0 |
| true | true | 1518 | 0 | 0 | 0 |

*Tabel 17. Confusion matrix - ALS ingevoerde teksten DAN omzeilde planningsdatum*

De met ARM-algoritmes gevonden referentiewoorden en de met TM gedetecteerde bijbehorende teksten geven ALS *ingevoerde teksten* DAN *omzeilde data* relaties met een hoge betrouwbare voorspelling van de workarounds, waarbij ook tikfouten van de gebruiker gedetecteerd worden.

# 5. Discussie, conclusies en aanbevelingen

We reflecteren de resultaten van het onderzoek en leggen verbanden met de literatuur en onderzoeken van de afstudeerkring, waarna we de conclusies geven op de deelvragen en aanbevelingen verstrekken voor de praktijk en verder onderzoek.

## 5.1. Discussie

Met TM zijn referentiewoorden die omzeilde data aanwijzen in de tekst te extraheren en met ARM zijn de relaties tussen deze referentiewoorden en omzeilde data te detecteren. Op basis daarvan kunnen we de ingevoerde workaround teksten herleiden. Deze resultaten geven antwoord op de hoofdvraag waarbij we concluderen dat TM en ARM-algoritmes betrouwbaar in staat zijn om de meest voorkomende workaround teksten te detecteren, waarmee gebruikers data-invoer omzeilen.

Deze workaround teksten met overgeslagen data-invoer geven kennis over waar en hoe data is omzeild, zodat we in afstemming over de motieven van gebruikers het proces of systeem kunnen herontwerpen (Van de Weerd, Vollers, Beerepoot, & Fantinato, 2019). Deze detectie met data science zal de weerbaarheid en veerkracht van het informatiesysteem verbeteren, wat aansluit op de doelstelling van de Resilience Mining afstudeerkring. In breder perspectief dragen we bij het achterbleven onderzoeksgebied van workarounds (Alter, 2014).

Het frequent voorkomen van referentiewoorden in tekst is een effectief kenmerk om de workarounds te detecteren, alleen de hoofdletters en signaaltekens (Huuskonen & Vakkari, 2013) zijn door beperkingen in TM niet bruikbaar. We vermoeden dat deze signaalinformatie weinig bijdraagt, omdat zonder deze kenmerken de workarounds ook te detecteren zijn, wat overeen komt met de beperkte correlatie die afstudeerder Ten Cate (2020) constateert.

De template en data science aanpak van shadow-IT van Fürstenau & Rothe (2014) is bruikbaar in het deelgebied workarounds. De mate van voorkomen van workarounds is lager dan het routinematige karakter doet verwachten (Röder, Wiesche, & Schermann, 2014) en dit sluit ook aan op de praktijk bevindingen van afstudeerder Spronk (2020), waarbij workarounds maar 2 tot 3% voorkomen. Doordat referentiewoorden niet exclusief zijn voor workarounds, zijn analyses per deelverzamelingen potentieel omzeilde data nodig wat afwijkt van de standaard data-analyse stappen (Kobayashi, Mol, Berkers, Kismihok, & Den Hartog, 2018).

Omdat TM gericht is op proza in documenten en niet op additionele informatie in tekst is de dimensionality reduction niet toereikend (Kobayashi, Mol, Berkers, Kismihok, & Den Hartog, 2018). Het filteren van hoge frequentie door normalisatie van de verdeling lijkt goed toepasbaar en vraagt om externe validatie. De aggregatie van unieke waarden en herleiden van signaalinformatie-structuren in teksten vraagt om verdiepende technologie. Door deze tekortkoming is beperkte taalkundige filtering nodig en is onderzoek voor het automatiseren van de tekstmanipulatie gewenst.

## 5.2. Reflectie

Met de filtering verfijnen we de detectie van workarounds naar bijna 100% wat vraagtekens oproept over de kwaliteit van de gegeneerde data. Daarentegen hebben we één waarheidscenario niet gedetecteerd door de lage frequentie. De complexiteit van varianten van referentiewoorden, die ook gebruikt zijn voor niet workarounds en de verschillende

tikfouten in tekst, zijn voldoende indicatie dat de data realistisch is. Wel zijn de workarounds eenzijdig gericht op één soort omzeilde data, zodat onderzoek naar andere praktijk situaties gewenst is om de methode extern te valideren.

De FPGrowth en Apriori algoritmes verschillen minimaal en het Tertius algoritme geeft geen workaround relaties, zodat de resultaten niet te combineren zijn en herontwerp van de experimenten nodig was. De analyse van deelverzamelingen, de clustering, de normalisatie van de verdeling en detectie van combinatiewoorden is door voortschrijdend inzicht ontstaan zonder draagvlak vanuit de literatuur. Dat referentiewoorden een *anker* zijn in de teksten en de potentieel omzeilde data een *sleutel*criteria is, hebben we laat ontdekt. Verdiepend onderzoek naar datum, nummer en code structuren en signaalinformatie is daardoor uitgebleven wat we voor vervolgonderzoek aanbevelen.

## 5.3. Conclusies

De conclusies hebben we samengevat door de drie deelvragen te beantwoorden.

*Hoe toepasbaar is TM voor het detecteren ingevoerde tekst die data omzeild?*
TM is geschikt om referentiewoorden te detecteren zoals *plannen* in de tekst *plannen op: 011280!,* waarbij deze tekst gebruikt is om de structurele datum invoer *01-12-80* te omzeilen. Het ontbreekt aan algoritmes om signaaltekens zoals "*:!"* erbij te voegen en ook om vooraf de omzeilde data in de vorm van nummers, codes of datums te extraheren uit de tekst. De woorden zijn op basis van de frequentie in teksten op significantie te filteren en bevatten de referentiewoorden van de workarounds. Ze onderscheiden zich door de normalisatie van de verdeling, zodat een beperkt aantal overblijft ten opzichte van een grote groep met veel sporadische en fout ingetikte woorden.

*Hoe toepasbaar is ARM om een relatie naar omzeilde data te detecteren?*
De FPGrowth en Apriori algoritmes zijn in tegenstelling tot het Tertius algoritme geschikt om relaties tussen referentiewoorden zoals *afleveren* in de tekst te verbinden met een *planningsdatum,* die door de gebruiker bij de invoer is overgeslagen. Deze relaties detecteren een workaround structuur waarbij de gebruiker de tekst invoert zoals *020860 : afleveren*. Omdat woorden zoals *afleveren* ook gebruikt worden voor andere informatie in een tekst voorspellen de relaties de *individuele* workarounds slecht.

Om de mogelijke invoer te bepalen die de gebruiker heeft overgeslagen, is een betrouwbare database nodig of kennis van het systeem. De ALS *referentie woord* DAN *potentiele omzeilde data* relaties detecteren we *alleen* per gebalanceerde deelverzameling potentieel omzeilde data, zoals de verzameling opdrachten waar de *planningsdatum* is overgeslagen. Bij het selecteren van *significante* relaties is de betrouwbaarheid de onderscheidende factor en is het K-Means algoritme geschikt om het cluster met een beperkt aantal betrouwbare relaties te detecteren.

*Welke kennis geven de relaties tussen ingevoerde tekst en omzeilde data?*
Omdat we met FPGrowth en Apriori betrouwbare referentiewoorden zoals *plannen* detecteren, hebben we een anker in de tekst om te herleiden wat de gebruiker *ervoor* of *erna* heeft ingevoerd. Hiermee vinden we ALS *ingevoerde teksten* DAN *omzeilde data* relaties, die met een hoge betrouwbaarheid de workarounds detecteren. Deze teksten bevatten de nummers, codes of datums die de omzeilde data zijn of geven de voor- en tussenvoegsels die daarnaar verwijzen.

Om de ingevoerde teksten te herleiden is een taalkundige selectie nodig is of ze verwijzen naar omzeilde data. Deze filtering is beperkt door verwijdering van vervuiling met normalisatie op verdeling en door de aggregatie van waarden. We lopen tegen de grenzen aan van de regular expression tekstmanipulatie, waardoor maatwerk programmatuur nodig is.

## 5.4. Aanbevelingen voor de praktijk

Op basis van de experimenten stellen we de volgende stappen voor om in de praktijk de workarounds in teksten te detecteren die data omzeilen:

*Voorbereiding:*

1. Genereer de template *omzeilde data structuren* vanuit het datamodel
2. Label met SQL de *potentiele omzeilde data* in de tabellen op basis van de template
3. Maak een *gebalanceerde deelverzamelingen* per *soort* potentieel omzeilde data

*Per deelverzameling:*

1. Genereer *tokens* uit de teksten en gebruik *BTO indexing*
2. Filter de tokens met de *laagste frequentie* eruit door normalisatie
3. Genereer de relaties met FPGrowth met de *tokens* en *potentieel omzeilde data*
4. Selecteer de enkelvoudige ALS *token* DAN *potentiele omzeilde data* relaties
5. Selecteer met K-means *clustering* de relaties met de *hoogste betrouwbaarheid*
6. Genereer *combinatietokens* die de *ingevoerde tekst* vormen met regular expression:
    a. Label de tokens van de relaties (5) als *referentiewoorden*
    b. Koppel de woorden of getallen *voor of na de referentiewoorden* in de tekst
    c. *Aggregeer* de woorden of getallen die *data* zijn in de tekst met *sleutelwoorden*
7. Filter de combinatietokens (teksten) met *laagste frequentie* eruit door normalisatie
8. Verfijn de ingevoerde teksten op taalkundige verwijzing naar *omzeilde data*

Deze stappen resulteren in ALS *ingevoerde teksten* DAN *omzeilde data* relaties, die kennis geven over de workaround teksten, zodat evaluatie over de gebruikersmotivatie mogelijk is. Door de dimensionality reduction en deelverzamelingen is de verwerkingstijd acceptabel. Zo is de verwerkingstijd van de 1,5 miljoen opdrachten met 50% gevulde teksten ongeveer 45-60 minuten op een PC.

## 5.5. Aanbevelingen voor verder onderzoek

De methodiek om workaround teksten die data omzeilen te detecteren, vraagt om validatie in meerdere praktijk cases (Saunders, Lewis, & Thornhill, 2016) met aansluitend onderzoek of de gedetecteerde workarounds inzicht geven in de ontwerpverbeteringen (Röder, Wiesche, & Schermann, 2014).

Door te onderzoeken of de methode ook valide is bij verschillende structuren van datums, nummers en codes in tekst, stellen we voor om tekstmanipulatie- en aggregatietechnieken te onderzoeken om de woorden rond het referentiewoord breder en automatisch te herleiden en de bijdrage van signaalinformatie te toetsen. Dit zal vragen om onderzoek naar additionele technieken met programmeertalen en functies aangezien de mogelijkheden van RapidMiner beperkt is.

# Dankwoord

Ik wil mijn afstudeerbegeleider Lloyd Rutledge bedanken voor de vele feedback die mijn onderzoek focus heeft gegeven op probleemstelling en structuur. Ook bedank ik meelezer Guy Janssens die aanzette tot samenvattende conclusies en een breder perspectief.

De wijze waarop de Open Universiteit deze begeleiding inricht, geeft voor mij een hoog kwaliteitsrendement, met name gevormd door het regelmatig bij elkaar komen van onze begeleider en afstudeer kringleden Ruben ten Cate, José Huisman, Maarten Koskamp, Thomas Sandfort, Patrick van der Spoel en Janneke Spronk. Elkaars inzichten, feedback, discussies en ook tegenslagen brachten het individuele onderzoek naar een teaminzet.

De beschikbare tijd van mijn werkgever Windesheim Hogeschool en de ruimte die mijn vrouw Alice van der Kamp heeft gegeven om in de afgelopen jaren mij soms dagen terug te laten trekken, maakte het mogelijk om deze studie en dit afsluitende onderzoek te verrichten. Het is een persoonlijk voorrecht wat ik bijzonder waardeer.

Door dit onderzoek heb ik veel kennis opgedaan in het samenstellen van een academisch werk, wat bijdraagt aan de doelstelling om mijn analytisch vermogen te verbeteren.

Jan van Rouwendal

# Referenties


Alter, S. (2014). Theory of Workarounds. *Communications of the ACM (34:1)*, 1041–1066.

Arora, J., Shelza, & Rao, S. (2013). An Efficient ARM Technique for Information Retrieval in Data Mining. *International Journal of Engineering Research & Technology (IJERT)*.

Collins, S., Fred, M., Wilcox, L., & Vawdrey, D. (2012). Workarounds Used by Nurses to Overcome Design Constraints of Electronic Health Records . *Partners HealthCare, Wellesley, MA, USA*.

Fürstenau, D., & Rothe, H. (2014). Shadow IT Systems: Discerning the Good and the Evil. *Twenty Second European Conference on Information Systems*.

Huuskonen, S., & Vakkari, P. (2013). I Did It My Way": Social Workers as Secondary Designers of a Client Information System. *Information Processing & Management (49:1)*, 380–391.

Jović, A., Brkić, K., & Bogunović, N. (2014). An overview of free software tools for general data mining. *37th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*.

Kobayashi, V., Mol, S., Berkers, H., Kismihok, G., & Den Hartog, D. (2018). Text Mining in Organizational Research. *Organizational Research Methods 2018, Vol. 21(3)*, 733-765.

Kopper, A., & Westner, M. (2016). Towards a Taxonomy for Shadow IT. *Americas Conference on Information Systems*.

Kulkarni, M., & Kulkarni, S. (2016). Knowledge Discovery in Text Mining using Association Rule Extraction. *International Journal of Computer Applications (0975 – 8887) Volume 143 – No.12,*.

Röder, N., Wiesche, M., & Schermann, M. (2014). Situational Perspective on Workarounds in It-Enabled Business Processes: A Multiple Case Study. *ECIS 2014 Proceedings*.

Saunders, M., & Thornhill, A. (1997). Research Methods for Business Students. Edinburth Gate: Pearson Education Limited.

Saunders, M., Lewis, P., & Thornhill, A. (2016). Research Methods for Business Students. In *Research Methods for Business Students.* Pearson Education.

Spronk, J. (2020). *Thesis: Mining for workarounds in text fields with clustering algorithms.* Netherlands: Open Universiteit.

Ten Cate, R. (2020). *Thesis: Detecting shadow IT in free text fields using text mining and classification.* Netherlands: Open Universiteit.

Van de Weerd, I., Vollers, P., Beerepoot, I., & Fantinato, M. (2019). *Workarounds in retail work systems: prevent, redesign, adopt or ignore?* ECIS 2019.